LASH's Past, Present and Future

LASH Audio Session Handler

- The Finnish Centre for Open Source Solutions (COSS) showed us love

- The Finnish Centre for Open Source Solutions (COSS) showed us love
- The results are:
  - Better (free)desktop integration via D-Bus

- The Finnish Centre for Open Source Solutions (COSS) showed us love
- The results are:
  - Better (free)desktop integration via D-Bus
  - A new client API

Summercode 2008

- The Finnish Centre for Open Source Solutions (COSS) showed us love
- The results are:
  - Better (free)desktop integration via D-Bus
  - A new client API
  - Lots and lots of refactoring

Summercode 2008

- Complete overhaul of socket-based communication infrastructure in favor of D-Bus

LASH, JACK, and D-Bus

- Complete overhaul of socket-based communication infrastructure in favor of D-Bus
- LASH client library talks to LASH server over the Session Bus

LASH, JACK, and D-Bus

- Complete overhaul of socket-based communication infrastructure in favor of D-Bus
- LASH client library talks to LASH server over the Session Bus
- LASH server communicates with JACK server over the Session Bus

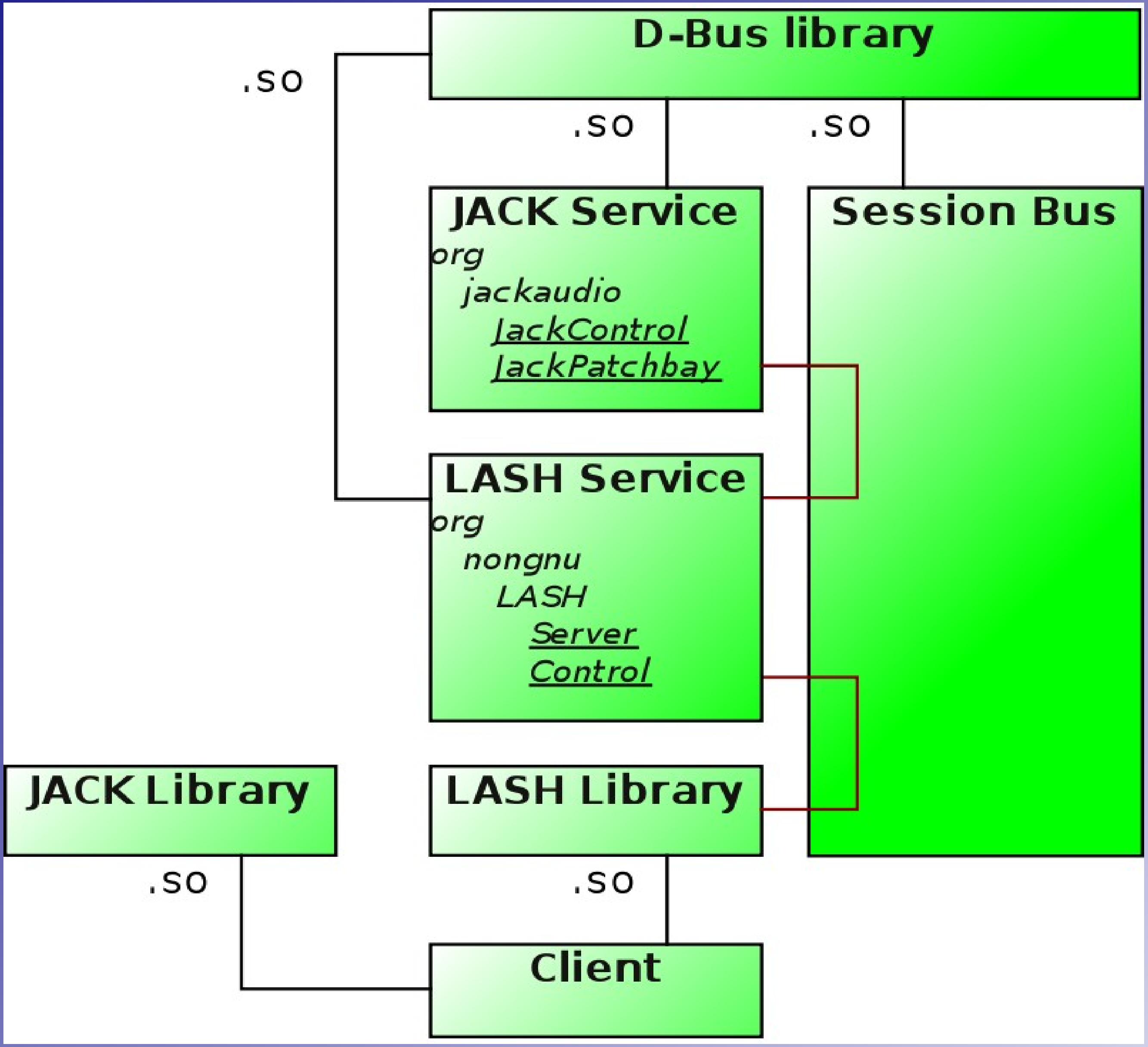
LASH, JACK, and D-Bus

- Complete overhaul of socket-based communication infrastructure in favor of D-Bus
- LASH client library talks to LASH server over the Session Bus
- LASH server communicates with JACK server over the Session Bus
- LASH (as well as JACK) controllable using any D-Bus-browser

LASH, JACK, and D-Bus

- Complete overhaul of socket-based communication infrastructure in favor of D-Bus
- LASH client library talks to LASH server over the Session Bus
- LASH server communicates with JACK server over the Session Bus
- LASH (as well as JACK) controllable using any D-Bus-browser
- Audio applications still link normally against libjack.so and liblash.so

LASH, JACK, and D-Bus



LASH, JACK, and D-Bus

```
// Parse args
lash_args_t *args = lash_extract_args(&argc, &argv);
```

Current API

```
// Parse args
lash_args_t *args = lash_extract_args(&argc, &argv);

// Create client
int flags = LASH_Config_Data_Set;
lash_client_t *client = lash_init(args, “Class”, flags,
                                   LASH_PROTOCOL(2, 0));
```

Current API

```
// Parse args
lash_args_t *args = lash_extract_args(&argc, &argv);

// Create client
int flags = LASH_Config_Data_Set;
lash_client_t *client = lash_init(args, “Class”, flags,
                                  LASH_PROTOCOL(2, 0));

// Send client name
lash_event_t *event =
    lash_event_new_with_type(LASH_Client_Name);
lash_send_event(client, event);
```

Current API

```
// Send JACK client name
lash_jack_client_name(client, “JackName”);
```

Current API

```
// Send JACK client name
lash_jack_client_name(client, “JackName”);

// Send ALSA client ID
lash_alsa_client_id(client, snd_seq_client_id(alsa_seq));
```

Current API

```
// Send JACK client name
lash_jack_client_name(client, “JackName”);

// Send ALSA client ID
lash_alsa_client_id(client, snd_seq_client_id(alsa_seq));

// Process
while (1) {
    if ((event = lash_get_event(client)) == <…>) {
        <...>
    }
    lash_event_destroy(event);
}
```

Current API

```
// Create client
int flags = LASH_Config_Data_Set;
lash_client_t *client = lash_client_open(“Class”, flags,
                                          argc, argv);
```

New API

```
// Create client
int flags = LASH_Config_Data_Set;
lash_client_t *client = lash_client_open(“Class”, flags,
                                          argc, argv);

// Send ALSA client ID
lash_alsa_client_id(client, snd_seq_client_id(alsa_seq));
```

New API

```
// Create client
int flags = LASH_Config_Data_Set;
lash_client_t *client = lash_client_open(“Class”, flags,
                                         argc, argv);

// Send ALSA client ID
lash_alsa_client_id(client, snd_seq_client_id(alsa_seq));

// Set callbacks
lash_set_save_data_set_callback(client, save_cb, NULL);
lash_set_load_data_set_callback(client, load_cb, NULL);
lash_set_quit_callback(client, quit_cb, NULL);
```

New API

```
// Process
while (1) {
    lash_wait(client);
    lash_dispatch(client);
}
```

New API

```
// Create controller
lash_client_t *client = lash_client_open_controller();

// Set control callback
lash_set_control_callback(client, ctrl_cb, NULL);

/* Control methods include:
   - lash_control_load_project_path(client, “/path”)
   - lash_control_save_project(client, “project”);
   - lash_control_close_project(client, “project”);
   - etc. */
```

New API

- New API intentionally looks much like JACK's

New API highlights

- New API intentionally looks much like JACK's
- Arguments extraction unnecessary

New API highlights

- New API intentionally looks much like JACK's
- Arguments extraction unnecessary
- LASH client name set by user, not the client

New API highlights

- New API intentionally looks much like JACK's
- Arguments extraction unnecessary
- LASH client name set by user, not the client
- JACK client identity auto-detected by lashd

New API highlights

- New API intentionally looks much like JACK's
- Arguments extraction unnecessary
- LASH client name set by user, not the client
- JACK client identity auto-detected by lashd
- No event object allocation/deallocation; everything happens in callbacks

New API highlights

- New API intentionally looks much like JACK's
- Arguments extraction unnecessary
- LASH client name set by user, not the client
- JACK client identity auto-detected by lashd
- No event object allocation/deallocation; everything happens in callbacks
- Work to be done, feedback to be gathered

New API highlights

See you at the LASH workshop!

->